Re-engineering Legacy Mission Scientific Software

Charles D. Norton

NASA Jet Propulsion Laboratory
High Performance Computing Systems Group
Observational Systems Division and
Center for Space Mission Information and Software Systems

Viktor K. Decyk

UCLA Department of Physics and Astronomy and
NASA Jet Propulsion Laboratory
High Performance Computing Systems Group
Observational Systems Division

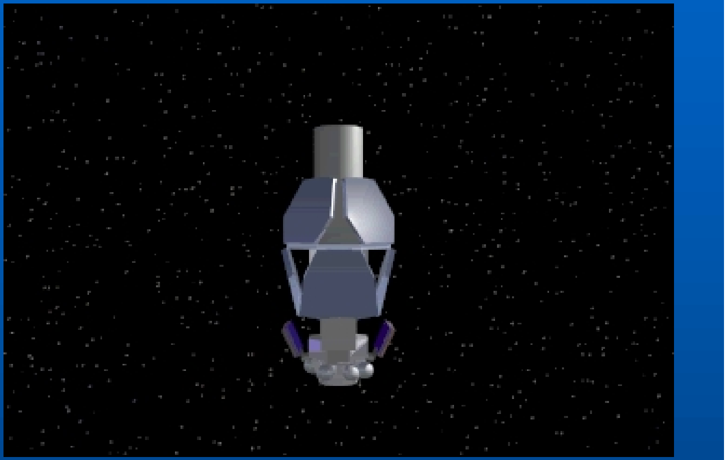
Modernizing Scientific Software

- **Legacy Software Has Value**
  - Generally well debugged with correct and trusted results
  - Actively used meeting end-user goals
- **Legacy Software Has Limitations**
  - Difficult to extend, modify, and support collaborative development
- **Rewriting Software Has Additional Costs**
  - Developing new verification/validation tests can be expensive
- **Abandon Legacy Code or Modernize It?**
  - If functionality is sound, legacy code can be wrapped in a modern interface where original code is mostly unmodified

Modernizing Scientific Software

- **CSMISS SET Activity**
  - Develop an incremental approach to re-engineer legacy systems
  - Examine how our methodology may benefit various JPL mission software projects
  - Demonstrate this for MACOS optical analysis software important to many NASA projects (D. Redding (385))

Software Re-engineering Benefits Optics Modeling for Mission Critical Projects



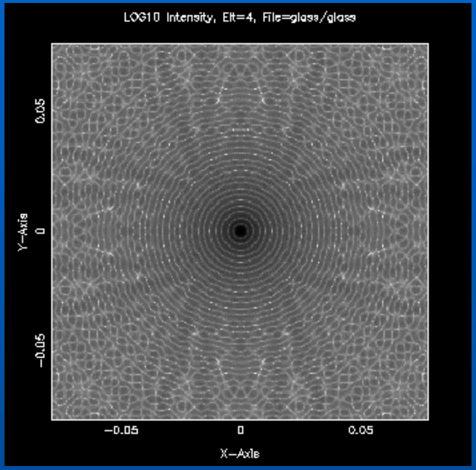
Next Generation Space Telescope

NASA Origin's program
NASA/GSFC/JPL

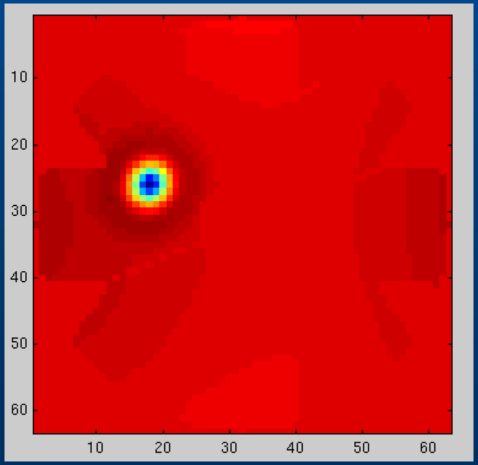
- **Current Results**
  - Methodology successfully applied & delivered to MACOS project
  - Meetings with JPL projects including future work involving the Navigation and Flight Mechanics Section

MACOS Overview

- **Modeling and Analysis for Controlled Optical Systems (MACOS/SMACOS)**
  - Developed by David Redding, and others, from Optical Systems Modeling Group (385)
  - Provides powerful optical analysis tools and a unique capability for system-level design tasks
- **Short List of Features**
  - Modeling optics on dynamic structures, deformable optics, and controlled optics
  - Efficient general ray-trace capabilities
  - Integrated support with other tools to create an end-to-end instrument system model



Modeling and Analysis of Controlled Optical Systems (MACOS) Program



MACOS Project Goals

- **Enhancement to benefit from modern software engineering techniques**
  - New features of Fortran 90/95 standard
  - Dynamic memory
  - Problem domain based design
  - Reorganization to promote collaborative development

Protect existing investment in software development and efficiency while benefiting from increased safety, organization, and extensibility

Technology

- **Fortran 90/95 Features Modernize Programming**

Modules
Encapsulate data and routines across program units

Interfaces
Verifies argument types in procedure calls

Array Syntax
Simplifies whole array, and array subset, operations

Use-Association
Controls access to module content

Derived Types
User-defined types supporting abstractions in programming

Pointers/Allocatable Arrays
Supports flexible/dynamic data structures

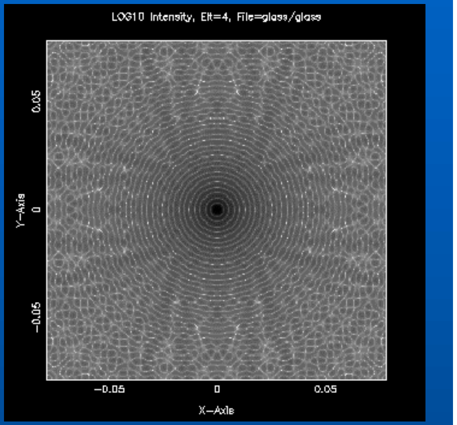
Backward compatible with Fortran 77

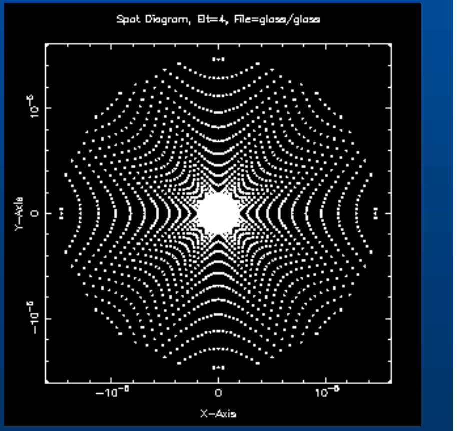
FOR MORE INFO...

Fortran 90 Programming. Ellis, Philips, & Lahey; Addison Wesley, 1994
http://www.cs.rpi.edu/~szymansk/oof90.html

Technology

- **Preserve existing software, yet transform for new development**
  - Create wrapper/interface layer to original code
  - Support abstraction-based programming via interfaces
  - Gradual/selective replacement of data structures
- **Benefits**
  - Software remains in use during modification
- **Not automatic, but largely mechanical...**
  - Automation projects, with limited capabilities, underway

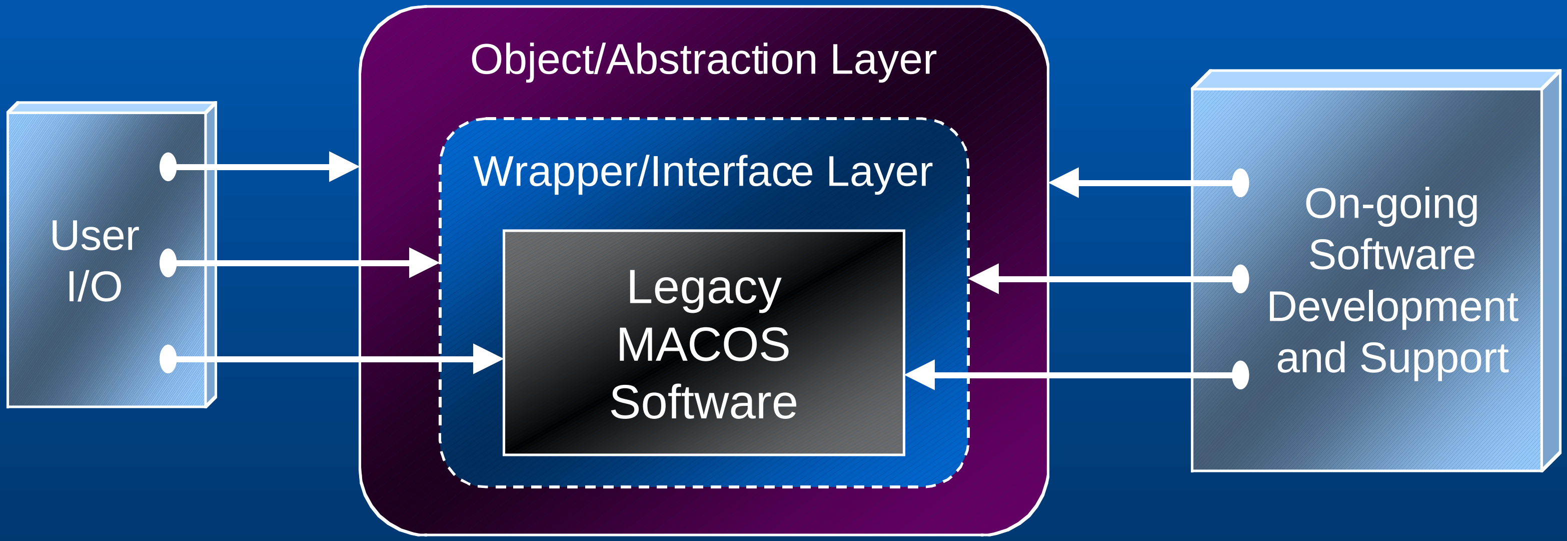




Important as more ambitious codes are developed and maintained

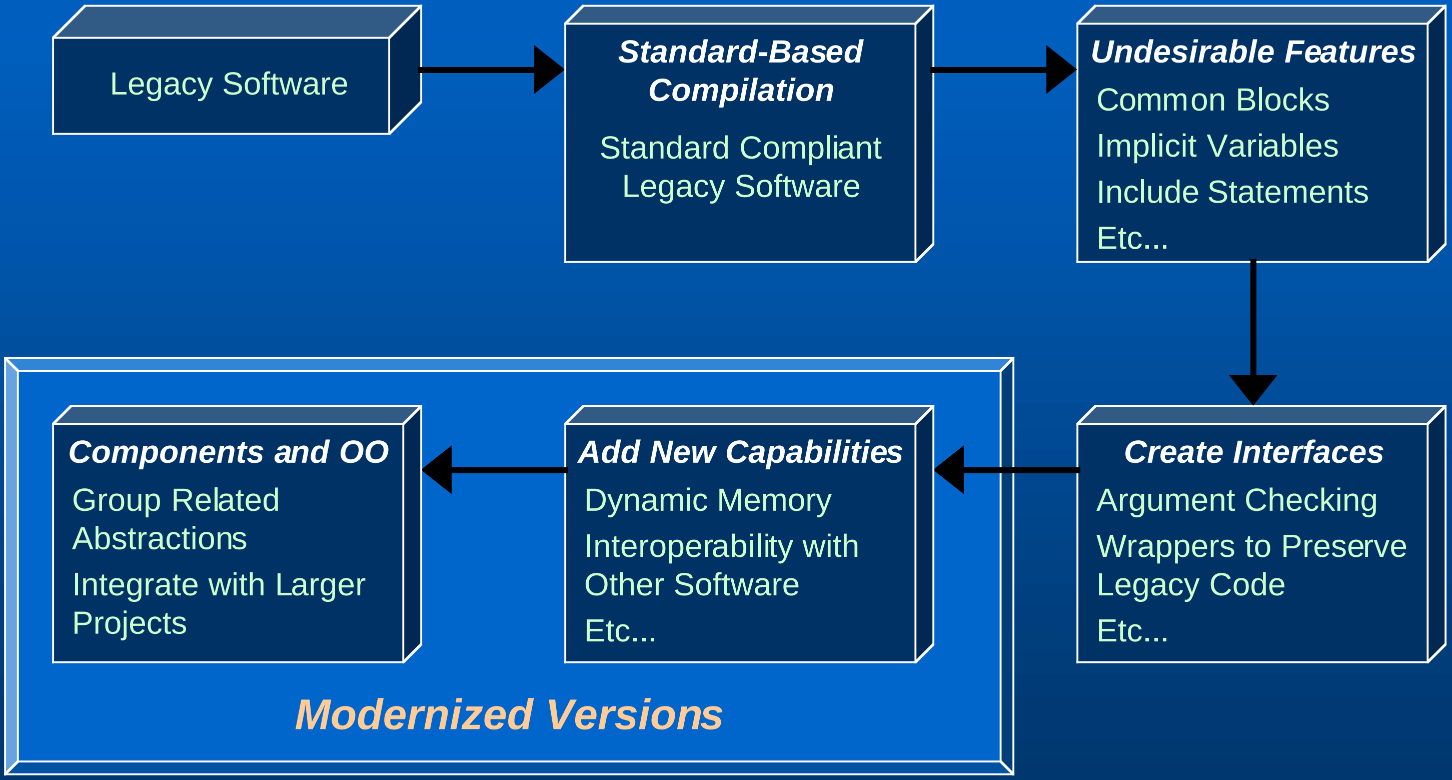
Modernization Process

- **Efficient interaction among MACOS, interfaces, abstraction layer, and user I/O**



Allows safe interaction with a legacy code

Modernization Process



Stages can vary based on original design of legacy software

Addressing Undesirable Features

- **Common Blocks**
  - Inhibit dynamic memory, exposure discourages code sharing, modification causes inadvertent errors

```
! Original COMMON Block in file
  common.inc
  real arg1(10,10), arg2(10,10)
  logical arg3
  integer arg4
  COMMON /BLOCK1/ arg1, arg2, arg3, arg4
  SAVE /BLOCK1/

  subroutine foo()
  include ‘common.inc’
  ...
  end subroutine
```

```
! Modernized version in common.f90
  MODULE common_block1
   implicit none
   save
   real, dimension(10,10) :: arg1, arg2
   logical :: arg3
   integer :: arg4
  END MODULE common_block1

  subroutine foo()
  use common_block1
  ...
  end subroutine foo
```

Conversion to modules is straightforward and brings many benefits...

Building Standard Wrappers

- **Interfaces**
  - Simplifies argument lists
  - Performs argument checks
  - Supports wrappers preserving legacy code

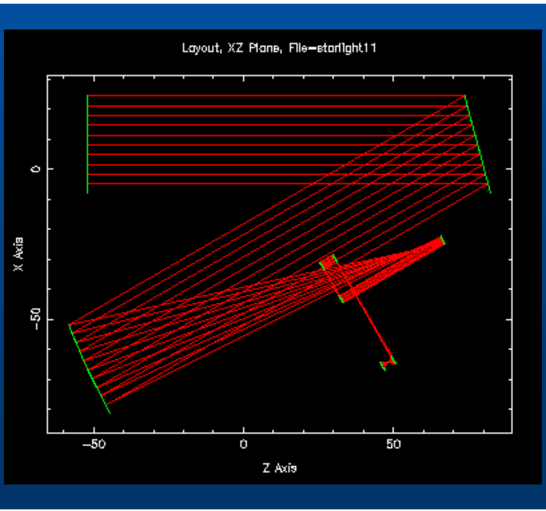
```
! Interface Module
  MODULE interface_module
   use common_block1
   implicit none
   save
   interface
     subroutine foof77(arg1, arg2, dim1)
       real, arg1(dim1, dim1)
       integer :: dim1
       logical :: arg2
     end subroutine
   end interface
  CONTAINS
  subroutine foof90(arg1, arg2)
   real, dimension(:,:) :: arg1
   logical :: arg2
     call foof77(arg1, arg2, size(arg1,1))
  end subroutine foof90
  END MODULE interface_module
```

- A simple example, but significant for complex procedures
- Modernized features can be applied at the interface/wrapper level

Building Standard Wrappers

- **Dynamic Memory**

```
! Legacy F77 include of COMMON data
   parameter (mdttl=128)
   integer nElt, RayID(mdttl, mdttl)
   COMMON /EltInt/ nElt, RayID,...
   SAVE /EltInt/
```



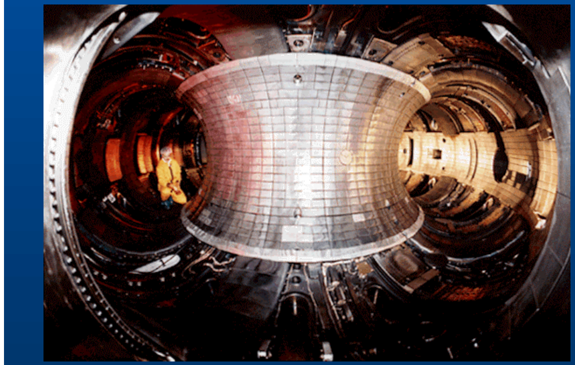
- Direct conversion of static data to dynamic data

```
! New Module for COMMON data with
  Dynamic Memory and Constructor
  Support
  MODULE elt_common
   implicit none
   save
   integer :: nElt, mdttl = 128
   integer, allocatable :: RayID(:,:)
  CONTAINS
! Constructor
   subroutine new_elt_common()
     allocate( RayID(mdttl, mdttl) )
   end subroutine new_elt_common
  END MODULE elt_common
! Dynamic Allocation from Main Program
  PROGRAM example
   use elt_common
   implicit none
   call new_elt_common()
   ...
  END PROGRAM example
```

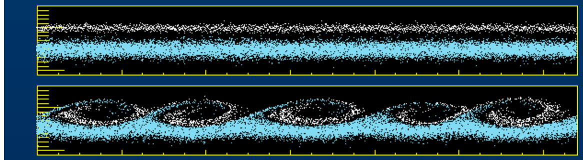
Building Standard Wrappers

- **Object-Oriented Design**
  - Legacy routine is wrapped by a simplified OO interface supporting dynamic components
  - Main program must call wrapper

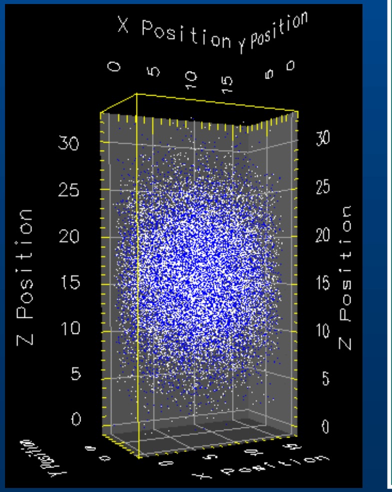
Tokamak Fusion Test Reactor (PPPL)



Beam-Plasma Instability



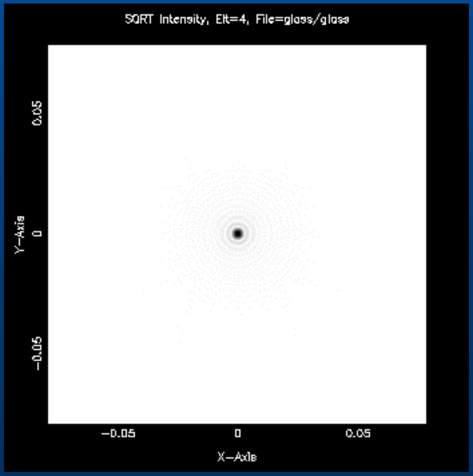
Free Expansion

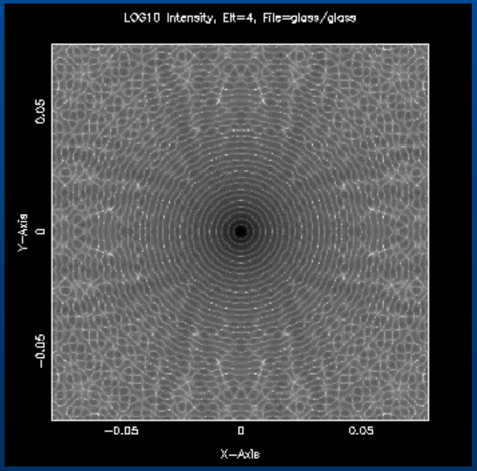


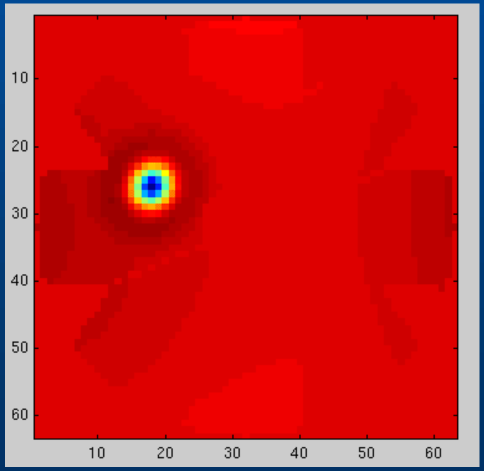
```
!  Encapsulation for Object-Design
  MODULE plasma_class
   type species
     real, dimension(:,:), pointer :: coords
     real :: charge_to_mass, kinetic_energy
   end type species
  CONTAINS
!  Using a legacy routine via an OO Wrapper
   subroutine w_push(particles, force, dt)
   type (species) :: particles
   real, dimension(:) :: force
   real :: dt, qbm, wke
   integer :: ndim, nparticles, nx
     ndim = size(particles%coords,2)
     nx = size(force)
     qbm = particles%charge_to_mass
     wke = particles%kinetic_energy
     call push(particles%coords, force, &
               qbm, wke, ndim, nparticles, &
               nx, dt)
   end subroutine w_push
  END MODULE plasma_class
```

Impact on MACOS

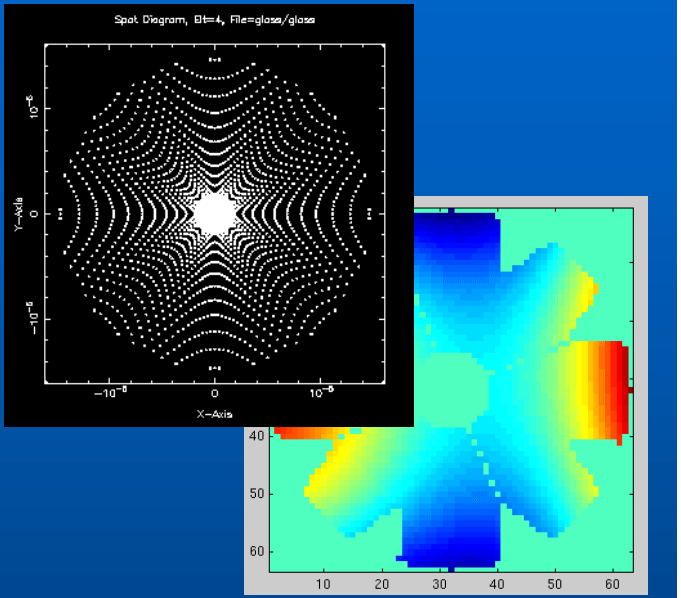
- **Improved Functionality**
  - System supports dynamic memory
  - Long-lasting subtle bugs corrected
  - More simple interfaces
  - Supports standardized language features







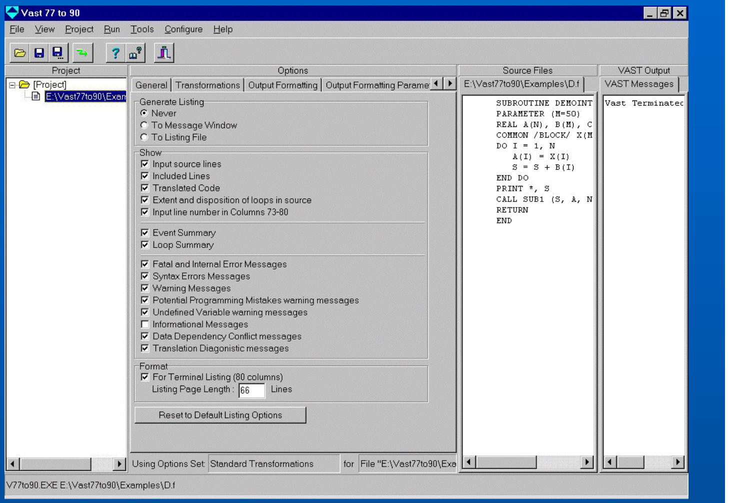
Modernization Experiences



- **Legacy Code Characteristics**
  - ~67,200 lines of Fortran 77
  - Distributed across ~60 files
  - Must interact with MATLAB and FFTW
  - ~765 Procedures
- **Modernized Code Characteristics**
  - ~3,500 additional lines of Fortran 90 (mainly interfaces)
    - 10 new modules, 28 required legacy interfaces, ~540 use statements replaced common block includes
  - .5 work year effort without participation of original authors
    - 2-3 times improvement expected when original authors participate
    - Speed/Performance comparable to original
  - The modernization process improved the code

Current Work Directions

- **Partial Automation**
  - Examine S/W tools for building interfaces automatically
- **Object-Oriented Interfaces**
  - Research how modernization process can add object-oriented concepts to legacy software
- **Define Software Architecture**
  - Separate GUI from physics
  - Extend concepts to other languages
- **Software Integration with NGST**
  - Moving modernized code into executive software that remotely drives the optical testbed hardware at GSFC



Generate Listing
● Never
○ To Message Window
○ To Listing File

Show
☑ Input source lines
☑ Included Lines
☑ Translated Code
☑ Extent and disposition of loops in source
☑ Input line number in Columns 73-80

☑ Event Summary
☑ Loop Summary

☑ Fatal and Internal Error Messages
☑ Syntax Errors Messages
☑ Warning Messages
☑ Potential Programming Mistakes warning messages
☑ Undefined Variable warning messages
☐ Informational Messages
☑ Data Dependency Conflict messages
☑ Translation Diagonistic messages

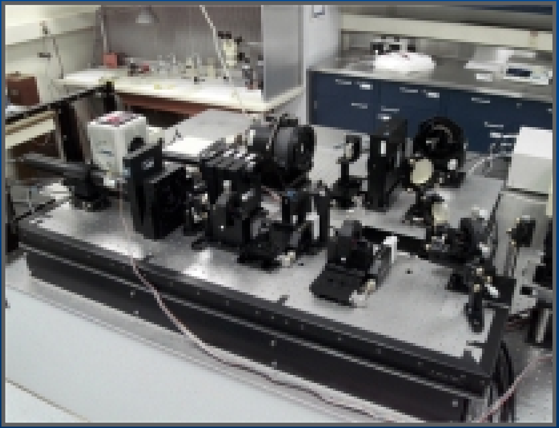
Format
☑ For Terminal Listing (80 columns)
Listing Page Length : 66 Lines

Using Options Set: Standard Transformations for File "E:\Vast77to90\Exa

```
      SUBROUTINE DEMOINT
      PARAMETER (M=50)
      REAL A(N), B(M), C
      COMMON /BLOCK/ X(M
      DO I = 1, N
         A(I) = X(I)
         S = S + B(I)
      END DO
      PRINT *, S
      CALL SUB1 (S, A, N
      RETURN
      END
```

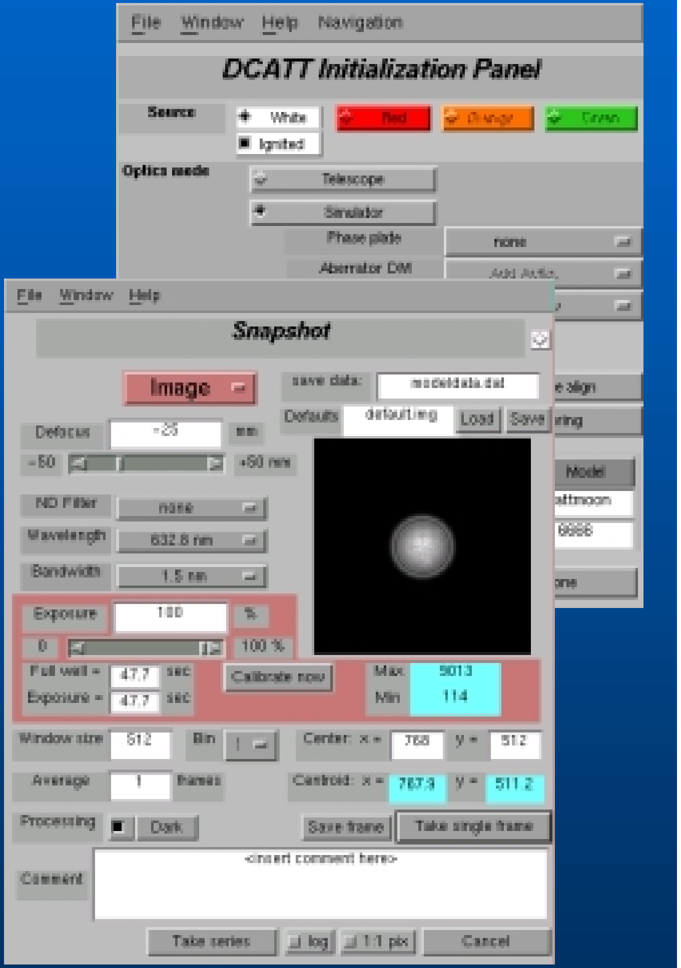
```
Vast Terminatec
```

V77to90.EXE E:\Vast77to90\Examples\D.f



Current Work Directions

- **Optical Systems Modeling Group**
  - Integration of new MACOS software with executive code to drive GSFC optical testbed remotely from JPL



DCATT Initialization Panel

Source: ● White ■ Ignited

Optics mode: ● Simulator

Phase plate: none

attmoon

6666

Snapshot

save data: modeldata.dat

Defaults: default.img

Defocus: -25 mm (-50 … +50 mm)

ND Filter: none

Wavelength: 632.8 nm

Bandwidth: 1.5 nm

Exposure: 100 % (0 … 100 %)

Full well = 47.7 sec

Exposure = 47.7 sec

Max: 5013

Min: 114

Window size: 512

Bin: 1

Center: x = 768 y = 512

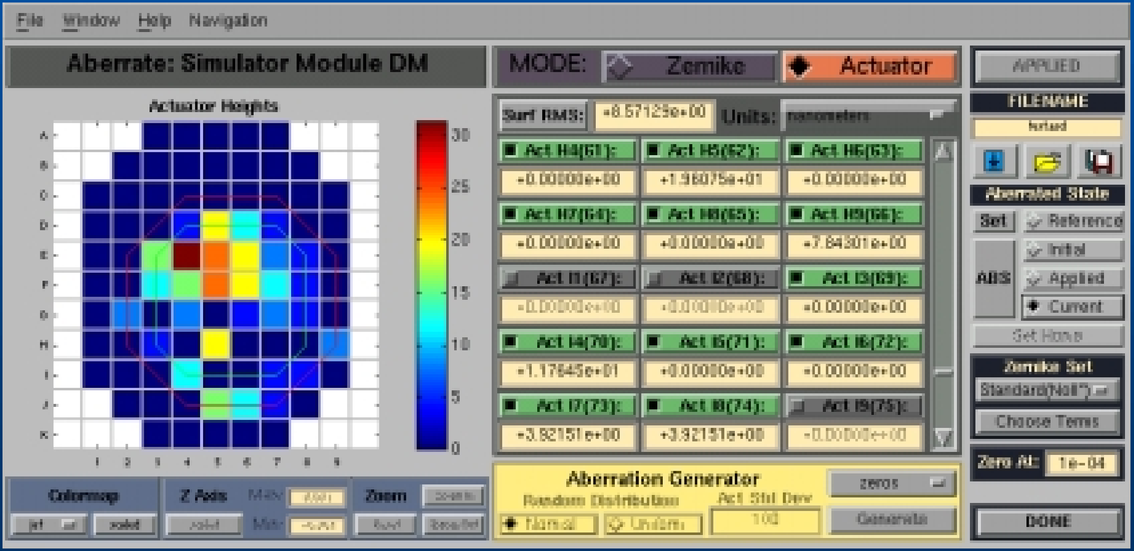
Average: 1 frames

Centroid: x = 767.9 y = 511.2

Processing: ■ Dark

Comment: <insert comment here>

□ log □ 1:1 pix



MODE: ◇ Zernike ◆ Actuator

Surf RMS: +8.57125e+00 Units: nanometers

■ Act H4(61):	■ Act H5(62):	■ Act H6(63):
+0.00000e+00	+1.96075e+01	+0.00000e+00
■ Act H7(64):	■ Act H8(65):	■ Act H9(66):
+0.00000e+00	+0.00000e+00	+7.84301e+00
□ Act I1(67):	□ Act I2(68):	■ Act I3(69):
+0.00000e+00	+0.00000e+00	+0.00000e+00
■ Act I4(70):	■ Act I5(71):	■ Act I6(72):
+1.17645e+01	+0.00000e+00	+0.00000e+00
■ Act I7(73):	■ Act I8(74):	□ Act I9(75):
+3.92151e+00	+3.92151e+00	+0.00000e+00

FILENAME: hirfaed

Aberrated State: Set ○ Reference; ABS ○ Initial ○ Applied ● Current

Zernike Set: Standard(Noll)

Zero At: 1e-04

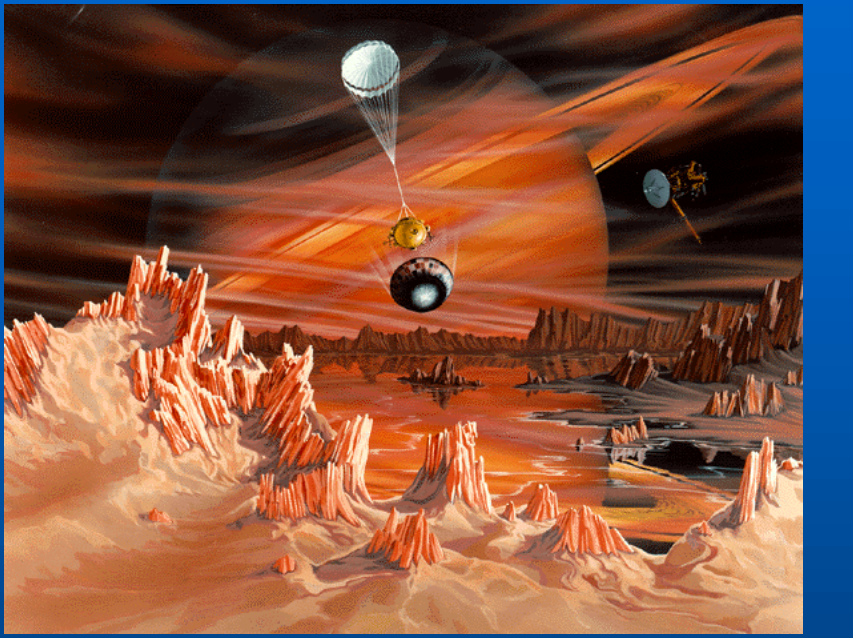
Aberration Generator: zeros

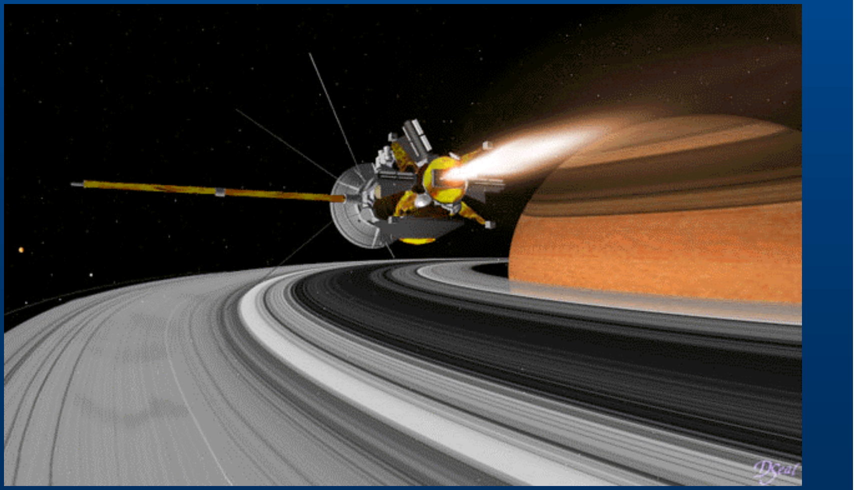
Random Distribution: ● Normal ○ Uniform

Act Std Dev: 100

Current Work Directions

- **Navigation Software Development Group**
  - Support over 6 MLOC in Fortran 77 and C
  - Require migration to F90/95 since F77 will be discontinued on current platform
  - Must support current, and new, missions without abandoning existing work
  - Cannot afford delays in schedule, but new development must continue
  - Successfully applied this process to parts of their code





Benefits of Re-engineering Methodology

- **Legacy codes still have value, but extending that functionality has become more important**
- **Modern codes require...**
  - Greater complexity and multiple authors
  - Dynamic features and flexible design
- **Build modern superstructure while code remains in use**
  - Data abstraction and information hiding are key to limiting exposure of unnecessary details
  - Modern language features reduce inadvertent errors
- **Wrappers can extend functionality**
  - Verify preconditions, measure performance, etc...

More Information

- **On the Web...**
  - http://hpc.jpl.nasa.gov/PEP/nortonc/csmiss.html
    - White Papers, FY 2000 Final Report, etc...
- **Acknowledgment**
  - Center for Space Mission Information and Software Systems
  - Observational Systems Division
    - High Performance Computing Group
    - Optical Systems Modeling Group
  - Navigation and Mission Design Section
    - Navigation Software Group